Solving Masked Jigsaw Puzzles with Diffusion Vision Transformers

Supplementary Material

6. Expanded Approach Description

We start this section by revisiting the mathematical formulation of conditional diffusion models, alongside an exploration of the self-attention mechanism and positional encoding inherent in transformers.

6.1. Conditional Diffusion Models

Diffusion models [10] are generative models characterized by forward and reverse diffusion processes. The forward process destroys the data from the true distribution $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ using a Markov chain to add Gaussian noise at each step: $q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$, where β_t is a small positive constant that represents the noise level, t is the diffusion step, and $\mathbf{I}$ is the identity matrix. Since the noise used at each step is Gaussian, $q(\mathbf{x}_t|\mathbf{x}_0)$ can be obtained in closed-form $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_0, (1-\alpha_t)\mathbf{I})$, where $\alpha_t = \prod_{s=1}^{t}(1-\beta_s)$. The reverse process denoises $\mathbf{x}_t$ to recover $\mathbf{x}_0$, and is defined as: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_\theta(\mathbf{x}_t, t)\mathbf{I})$, where μ_θ and σ_θ are approximated by a neural network. [10] has shown that this reverse process can be trained by solving the optimization problem $\min_\theta \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|_2^2$, where ϵ_θ is a trainable denoising function, and estimates the noise vector ϵ that was added to its noisy input $\mathbf{x}_t$.

Conditonal Diffusion Models Diffusion models are in principle capable of modelling conditional distributions of the form $p(\mathbf{x}|\mathbf{y})$, where $\mathbf{y}$ is a conditional input. In the context of conditional diffusion models, the model learns to predict the noise added to the noisy input given a set of conditions, including the time step t and the conditional inputs $\mathbf{y}$. The reverse process in this case is defined as: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y}) := \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t|\mathbf{y}), \sigma_\theta(\mathbf{x}_t, t|\mathbf{y})\mathbf{I})$. The conditional diffusion model learns a network ϵ_θ to predict the noise added to the noisy input $\mathbf{x}_t$ with

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0),\epsilon\sim\mathcal{N}(0,I),t,\mathbf{y}}[\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t, y)\|_2^2] \quad (1)$$

where $\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + (1-\alpha_t)\epsilon$, and $\mathcal{L}$ is the overall learning objective of the diffusion model.

6.2. Transformers and positional encoding

Transformers Recent advancements in Natural Language Processing (NLP) and Computer Vision (CV), exemplified by *Transformers* [7, 29], have garnered considerable success. In the realm of *transformers*, each element in a sequence or every patch in an image is typically embedded into a vector or token. These tokens traverse through an architecture composed of a stack of Self-Attention (SA) and Multi-Layer Perceptron (MLP) modules, with the SA mechanism standing out as their fundamental component. The SA module is designed to capture long-range interactions among three types of inputs: queries $\mathbf{Q}$, keys $\mathbf{K}$, and values $\mathbf{V}$, where the values are linearly combined according to the importance of each key representing each of the queries.

More concretely, let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D\times N}$, denote a set of N data tokens. Then, the queries, keys and values $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_N] \in \mathbb{R}^{d\times N}$, $\mathbf{K} = [\mathbf{k}_1, \ldots, \mathbf{k}_N] \in \mathbb{R}^{d\times N}$, and $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_N] \in \mathbb{R}^{d\times N}$, are computed through linear multiplication of the inputs with learnable weights denoted by $\mathbf{W}^Q \in \mathbb{R}^{D\times d}$, $\mathbf{W}^K \in \mathbb{R}^{D\times d}$, and $\mathbf{W}^V \in \mathbb{R}^{D\times d}$:

$$\mathbf{Q} = \mathbf{W}^Q\mathbf{X}, \ \mathbf{K} = \mathbf{W}^K\mathbf{X}, \text{ and } \mathbf{V} = \mathbf{W}^V\mathbf{X}.$$

Subsequently, a similarity matrix, or the attention map, is calculated using the dot-product between queries and keys. The normalized attention map, is then employed for the weighted aggregation of the values:

$$\mathrm{SA}(\mathbf{X}) = \mathbf{V}\mathrm{softmax}\left(\frac{\mathbf{K}^T\mathbf{Q}}{\sqrt{d}}\right) \quad (2)$$

where

$$\mathrm{softmax}\left(\frac{\mathbf{K}^T\mathbf{Q}}{\sqrt{d}}\right)_{i,j} = \frac{\exp(\mathbf{k}_i^T\mathbf{q}_j/\sqrt{d})}{\sum_i \exp(\mathbf{k}_i^T\mathbf{q}_j/\sqrt{d})} \quad (3)$$

An essential characteristic of SA is its lack of awareness of positional information in the input[32]. In other words, the output's content remains independent of the input order:

$$\mathcal{S}(\mathrm{SA}(\mathbf{X})) = \mathrm{SA}(\mathcal{S}(\mathbf{X})) \quad (4)$$

where $\mathcal{S}(\mathbf{X}) = [\mathbf{x}_{\pi_1}, \ldots, \mathbf{x}_{\pi_N}]$ denotes a shuffle operator on the tokens, with the left and right instances being the same permutation. Positional information can be introduced into the tokens by incorporating the use of positional encoding, which is added to the tokens before the SA and Multi-Layer Perceptron (MLP) layers.

As suggested in [12], the information utilized by SA can be categorized into three types: 1) relative positional-based attention, 2) absolute position-based attention, and 3) contents-based attention. For solving temporal jigsaw puzzles, where positional information might be less informative, it is natural to prioritize the model's focus on content-based information.

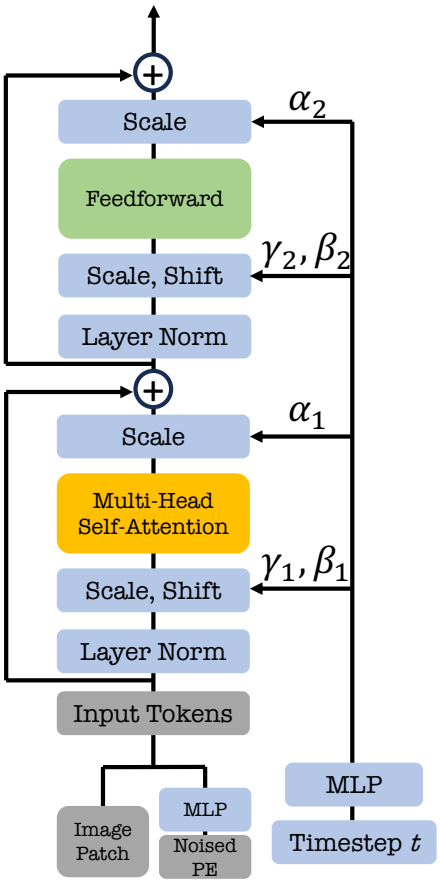

Figure 8. Diffusion transformer block for image jigsaw puzzles.

Table 6. Hyperparameters for our image diffusion transformer

Model	DiT
Layers	12
Hidden dimension	768
MLP size	3072
Heads	12
Patch size	16 / 17 / 60

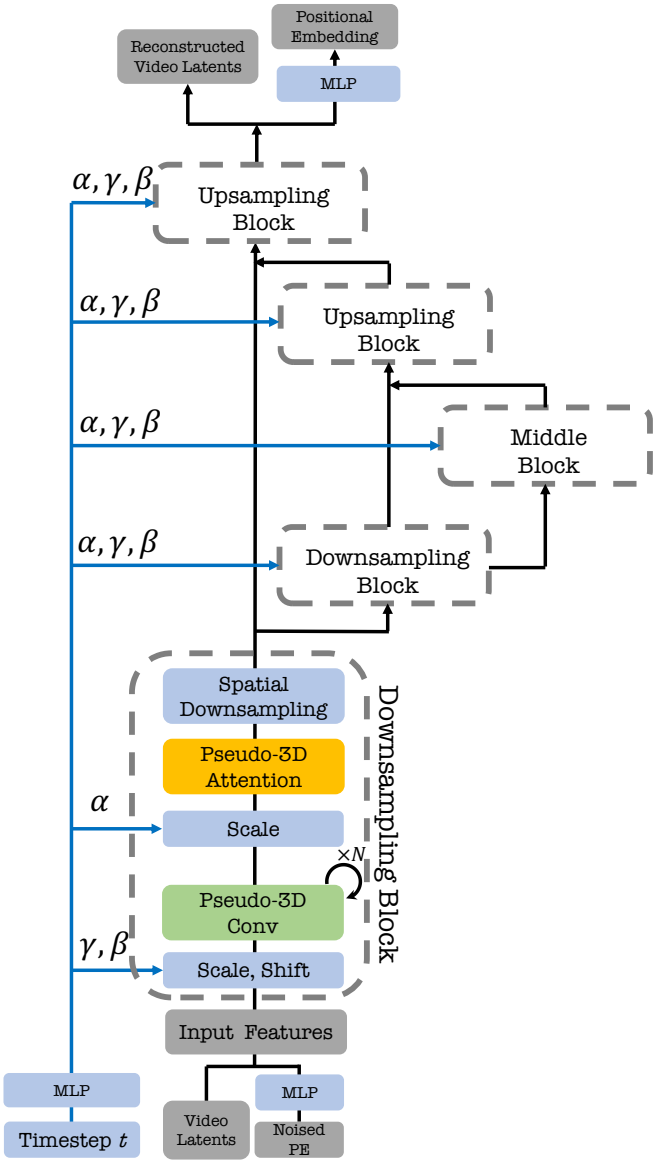

Figure 9. Illustration of video model that we use for solving video jigsaw puzzles.

7. Implementation Details

7.1. Image Experiments

Diffusion Vision Transformer architecture. We followed the standard ViT architecture, with extra Adaptive normalization layers added before and after MLP layers and multi-head attention layers, shown in Fig 8. Notably, our approach applies the diffusion model directly to image pixels rather than the latent space, providing a unique perspective on image understanding. In the context of image jigsaw puzzles without a gap, each image undergoes resizing and random cropping to a fixed size of 192×192 pixels. Within the DiT framework, the image is further segmented into 3×3 patches with 64×64 pixels resolution, subsequently embedded into 9 patch tokens. To explore the impact of eroded gaps in image jigsaw experiments, images are initially resized to 255×255 and then cropped into 3×3 patches with a resolution of 85×85 pixels. During training, a patch size of 64×64 is randomly selected, whereas in testing, a center crop method is employed to obtain image patches. The noised positional embedding undergoes processing through a Multilayer Perceptron (MLP) to align its dimension with that of the image tokens. Subsequently, the image tokens are added with the positional embedding tokens, forming an input that is fed into the diffusion transformers.

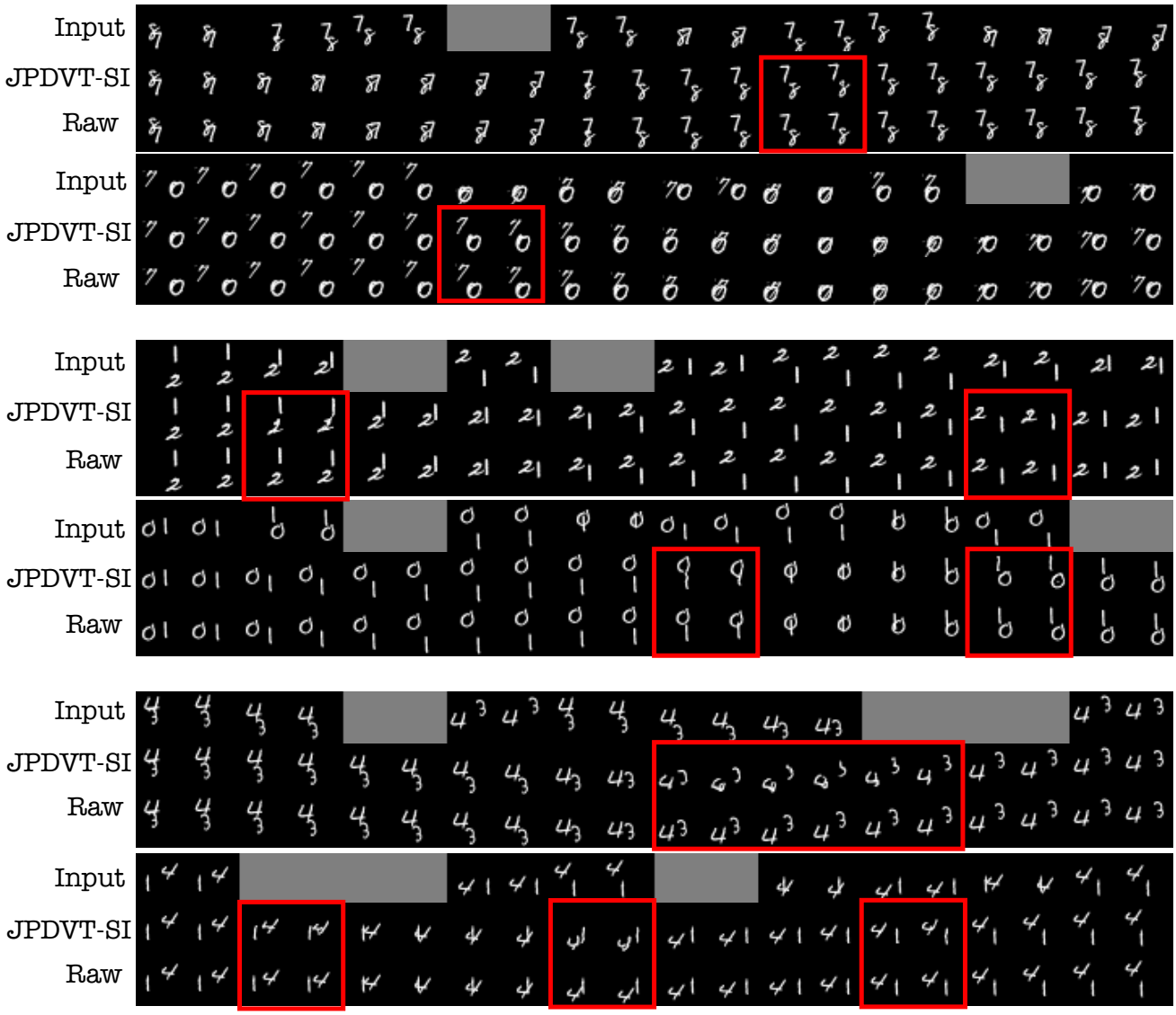

Figure 10. The reshuffling and inpainting results on 10 pieces *MovingMNIST* experiment with different numbers of missing pieces. Frames highlighted within red boxes showcase the inpainting capabilities of our model.

Our DiT models undergo training with a batch size of 256 over 300 epochs. A learning rate of 10^{-4} is employed, and the noised positional embedding undergoes diffusion through a linear schedule. The timestep T is consistently set at 1000.

Predicted position encodings. For a puzzle with N pieces, we generate N true position embeddings. Given a piece diffusion generated positional embedding, we find the closest true embedding (using L_2 distance) to assign the piece its final location. To avoid collisions, once a true position is used, it is removed as a candidate.

7.2. Video Experiments

Latent Diffusion Models. In our video models, we leverage the publicly available VAE encoder introduced in [22] to transition video frames from pixel space to latent space. Subsequently, we apply conditional diffusion models to the video latents. Each frame, initially sized at $256 \times 256 \times 3$, undergoes transformation into a latent feature map with dimensions $32 \times 32 \times 4$. The positional embedding undergoes processing through an MLP layer before integration with the video latents. Our chosen architecture, as proposed by [23], is composed of four downsampling blocks, a middle block, and four upsampling blocks. Within each

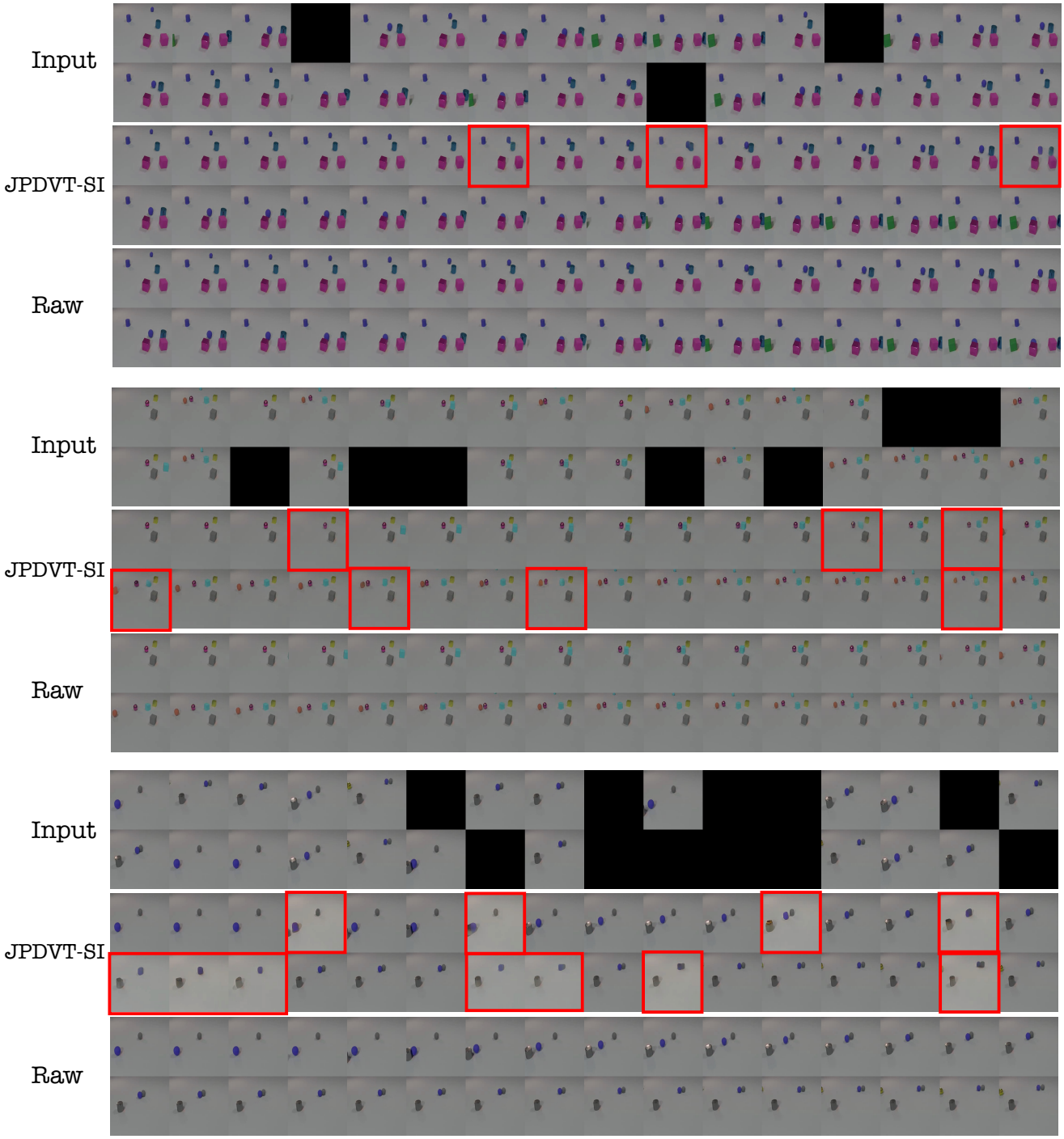

Figure 11. The reshuffling and inpainting results on 32 pieces *CLEVRER* experiment with different numbers of missing pieces. Frames highlighted within red boxes showcase the inpainting capabilities of our model.

block, there are multiple pseudo-3D convolutional layers, a pseudo-3D attention layer, and a spatial downsampling/upsampling layer. An illustrative depiction of the architecture is presented in Figure 9, and detailed hyperparameters for each block are meticulously documented in Table 7.

Our video models across various datasets were trained using a consistent batch size of 32. Each model under-

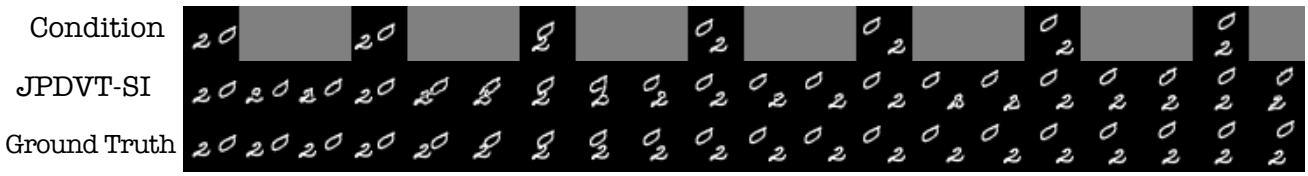

Figure 12. Video temporal super-resolution result.

Table 7. Hyperparameters in the downsampling blocks for our video models

#Block	1^{st}	2^{nd}	3^{rd}	4^{th}
Hidden dimension	64	128	256	512
Feature map size	32	16	8	4
# P-3D conv layers	3			
Attention head Dim	64			
# Attention heads	8			

went training for a total of 250,000 steps. We implemented a linear noise schedule, while keeping the total timestep, $T = 1000$. The learning rate for the training process was set to 10^{-4}. We trained our models on various datasets, each with a distinct temporal downsampling rate. Specifically, we applied downsampling factors of 1, 4, 2, and 2 for the MovingMNIST, CLEVRER, UCF101, and QST datasets, respectively. We used the same downsampling rate for the experiments run on the baselines.

In our experiments involving missing frames, we randomly sampled a varying number of missing elements, with the maximum set at 25% of the total pieces. Gaussian noise was introduced to both the positional encoding matrix $\mathbf{L}$ and the matrix representing missing pieces $\mathbf{E}^m$. In our loss function, we employed a ratio of 0.8 to 0.2 to balance the loss term of missing frames and the loss term of the positional encoding.

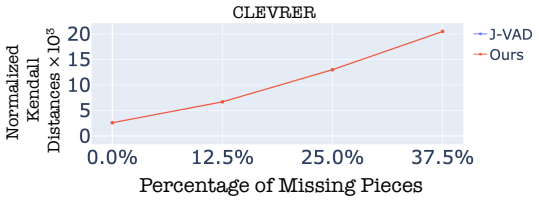

Figure 13. CLEVRER dataset with 32 pieces, each with 1 frame.

Additional results showcasing the outcomes of our video reshuffling and inpainting experiments are presented in Figures 10 and 11. These figures vividly illustrate our models' proficiency in concurrently learning both visual context and positional information. Figure 13 shows the results of Kendall distances of our proposed methods with different numbers of missing pieces with a total of 32 pieces on the CLEVRER dataset.

Table 8. Hyperparameters for experiments of J-VAD

Dataset	MMNIST	CLEVRER	UCF101
optimizer	AdamW		
learning rate	1e-4		
momentum	0.9		
weight decay	5e-4		
image size	32	64	112
batch size	64	32	8
epochs	300	200	200

Table 9. Hyperparameters for experiments on VCOP

config	MMNIST	CLVR	UCF	QST
optimizer	SGD			
learning rate	1e-3			
momentum	0.9			
weight decay	5e-4			
lr scheduler	RLRP			
batch size	256	16	16	4
image size	32	64	112	128

Implementation of baselines. We conducted baseline experiments utilizing publicly available code from the following papers: [6, 30, 33]. The specifics of the training configurations are outlined in Tables 8 and 9. In the case of experiments involving J-VAD [30] and scenarios with more than 10 pieces, the models were trained for 2,000 epochs.

8. Downstream Task: Temporal Super-resolution

The proposed framework can be directly applied to increase the temporal resolution of a given video, where the missing puzzle pieces are the intermediate frames. Fig. 12 shows an example where an input video with low sampling rate (top) was used to generate a video at a higher sampling rate (middle).